Lustre Development

Eric Barton
Lead Engineer, Lustre Group
Lustre Development

Agenda

• Engineering
  • Improving stability
  • Sustaining innovation

• Development
  • Scaling and performance
  • Ldiskfs and DMU

• Research
  • Scaling
  • Performance
  • Resilience
• Lustre – 257 KLOC

• Total of all in-tree linux filesystems – 471 KLOC
Engineering
Historical Priorities

Performance
Features
Stability
Engineering Priorities

- Stability
  - Reduce support incident rate
  - Reliable / predictable development
  - Address technical debt

- Performance & Scaling
  - Prevent performance regression
  - Exploit hardware improvements

- Features
  - Improve fault tolerance / recovery
  - Improve manageability
Engineering Knowledge

- ORNL
  - “Understanding Lustre Filesystem Internals”
- Lustre internals documentation project
  - Work in progress
  - Continuously maintained
- Subsystem map
- Narrative documentation
  - Asciidoc
- Api documentation
  - Doxygen
Engineering
Branch management

• Prioritize major development branch stability
  • Solid foundation
  • Reliable / early regression detection
  • Predictable / sustainable development

• Gatekeeper
  • Control landing schedule
  • Enforce defective patch backout
  • Influence patch size for inspection / test

• Git
  • Retained all significant CVS history
  • Single repository covers everything
  • Much easier backouts
Engineering Test

• Hyperion
  • 100s of client nodes
    • Multimount – simulate 1000s of clients
  • Multiple test runs weekly
  • Leverage much earlier in development cycle
• Daily automated testing
  • Results vetting
• Improved defect observability
  • See trends
  • Discern regular v. intermittent issues
  • Early regression detection
Engineering Process

- Clear release objectives
  - Manage risk – stability / schedule uncertainty
  - Release blockers defined by bug priority
- Bi-weekly builds
  - Formal test plans
  - Prioritize test issues
- Daily review
  - Engineering progress
  - Testing results
  - Issue priorities
Development Priorities

- Lustre 1
  - Maintenance
- Lustre 2
  - Stabilization
  - Performance
    - Eliminate regressions
    - Land improvements
  - Features
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Development Projects

- SMP scaling
  - Exploit multicore servers
  - Improve metadata throughput
- Platform portability
  - Extend OS-specific / portable layering to metadata
  - Formalize porting primitives
- Ldiskfs / DMU(ZFS) OSD
  - Pluggable storage subsystem
- HSM
- Clean server shutdown / restart
  - Simplify version interoperation / rolling upgrade
Development
Imperative Recovery

- Explicit client notification on server restart
Development
DMU performance

- Continued comprehensive benchmarking
- ZFS enhancements
  - Zero copy
  - Improved disk utilization
- Close cooperation with ZFS development team
Research Priorities

- Metadata Performance
- I/O Performance
- Numbers of clients
- Resilience and Recovery
Research

Numbers of Clients

• Currently able to accommodate 10,000s
• Next steps
  • System call forwarders - 10-100x
• Further steps
  • Caching proxies
  • Subtree locking
Research
I/O

• Initial NRS experiments encouraging
  • 40% Read improvement
  • 60% Write improvement

• Next steps
  • Larger scale prototype benchmarking
  • Exploit synergy with SMP scaling work

• Further steps
  • Global NRS policies
  • Quality of service
Research Metadata

• SMP scaling
  • Deeper locking / CPU affinity issues
• CMD Preview
  • Sequenced / synched distributed updates
  • Characterise performance
• Next Steps
  • Productize CMD Preview
• Further Steps
  • CMD based on epochs
Research
Resilience & Recovery

- $O(n)$ pinger overhead / detection latency
- Overreliance on client timeouts
  - $O(n)$ to distinguish server congestion from death
  - Include disk latency
  - Required to detect LNET router failure
- Over-eager server timeouts
  - Can’t distinguish LNET router failure from client death
- Recovery affects everyone
  - Transparency not guaranteed after recovery window expires
    - COS/VBR only partial solution
    - MDT outage disconnects namespace
Research
Resilience & Recovery

• Scalable health network design
  • Out-of-band communications
  • Low latency global notifications
  • Collectives: Census, LOVE reduction etc
  • Clear completion & network partition semantics
  • Self-healing

• Next steps
  • HN prototype
  • OST mirroring

• Further steps
  • Epoch based SNS
Lustre Development Summary

- Prioritize stability
  - Continued product quality improvements
  - Predictable release schedule
  - Sustainable development

- Continued innovation
  - Prioritized development schedule
  - Planned product evolution